UNITED STATES PATENT APPLICATION

for

THE ARCHITECTURE FOR DSR CLIENT AND SERVER DEVELOPMENT PLATFORM

Inventors:
Liang He
ChuanQuan Xie
Song Cui

prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90026-1026
(408) 720-8300

Attorney Docket No. 42390P12176

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL 61720 7195 US

Date of Deposit: January 24, 2002

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to BOX PATENT APPLICATION, Assistant Commissioner for Patents, Washington, D. C. 20231

Barbara Skliba

(Typed or printed name of person mailing paper or fee)

Barbara Skliba

(Signature of person mailing paper or fee)

1/24/02

(Date signed)

THE ARCHITECTURE FOR DSR CLIENT AND SERVER DEVELOPMENT PLATFORM

RELATED APPLICATION

[0001] This application is related to co-pending patent application no. ____________ entitled, "A Data Transmission System and Method for DSR Application over GPRS", filed January 24, 2002, which application is assigned to the assignee of the present application.

FIELD OF THE INVENTION

[0002] The present invention relates to distributed speech recognition (DSR) technology. More specifically, the present invention relates to the architecture design, implementation and optimization of a distributed speech recognition system.

BACKGROUND OF THE INVENTION

[0003] Currently, computer-supported communication via the Internet has, in a short time, become an important determinant of social life and a driving force of economic development in all developed and developing countries. In this field, global markets with huge potential for development have formed within a very short time out of a rapidly evolving process of innovation.

[0004] In order to meet the requirements of the explosive growth of the Internet technology, speech researchers have been putting a great deal of effort into integrating speech functions into Internet applications. For simple applications, voice playback and voice recording functions may

be sufficient. For complex applications, however, speech recognition functions are needed. For example, when it is possible for the users of mobile computing devices (such as personal digital assistants (PDA), wireless telephone, etc.) to access the Internet, especially, the World-Wide Web ("the Web") by voice, speech recognition must be done first. Speech recognition is a compute-intensive application. In order to balance computer load and to utilize the bandwidth of heterogeneous networks more effectively, distributed speech recognition (DSR) will undoubtedly become the best design choice.

[0005] Generally, there are three alternative strategies in the design of DSR architectures. The first strategy is server-only processing, in which all processing is done at the server side and the speech signal is transmitted to the server either through the Internet by using speech coding or via a second channel like telephone. Because all the recognition processing takes place at the server side, this approach has the smallest computational and memory requirements on the client, thereby allowing a wide range of client machines to access the speech-enabled application. The disadvantage of this approach is that users cannot access applications through a low-bandwidth connection.

[0006] The second conventional strategy is client-only processing, in which most of the speech processing is done at the client side and the results are transmitted to the server. The typical advantages are that a high-bandwidth connection is not required and recognition can be based on high-quality speech, because the sampling and feature extraction takes place at the client side. This approach is also less dependent on transmission channel and is

therefore more reliable. This approach, however, limits significantly the types of clients that the speech-enabled applications can support, because the client must be powerful enough to perform the heavy computation that takes place in the speech recognition process.

[0007] The third strategy is a client-server approach. In the client-server DSR processing model, front-end processing is done at the client side. Then, the speech features are transmitted to the server and finally the processing of the speech decoding and language understanding is performed at the server side.

[0008] At present, the client-server based distributed speech recognition approach has not been exploited. The client-server DSR approach requires a great deal of effort to integrate such a distributed speech recognition system based on client-server model into the Internet and wireless communication applications.

[0009] The difficulty of implementing an efficient client-server DSR architecture relates to diverse handheld device designs, network infrastructure complexity, network gateway or server diversity, and unstructured information content. To make such a distributed speech recognition system based on a client-server model happen in industry and the commercial market, the development efforts of many diverse areas are needed. As such, a comprehensive, practical, and efficient architecture for DSR in a client/server design has not been achieved in the conventional art.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The features of the invention will be more fully understood by reference to the accompanying drawings, in which:

[0011] Figure 1 illustrates a client-server based distributed speech recognition system.

[0012] Figure 2 illustrates the architecture for a DSR system development platform according to one embodiment of the present invention.

[0013] Figure 3 illustrates a method for accessing a document by speech based on the architecture for a DSR system development platform according to the one embodiment of the present invention.

[0014] Figure 4 illustrates the major components of the architecture shown in Fig.2 according to the one embodiment of the present invention.

[0015] Figures 5-7 illustrate the system synchronization design between DSR client module and DSR gateway module.

[0016] Figure 8 illustrates an example of the network environment in which the architecture for the DSR system development platform of the present invention may be applied.

DETAILED DESCRIPTION

[0017] In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be appreciated by one of ordinary skill in the art that the present invention shall not be limited to these specific details.

[0018] A distributed speech recognition system based on a client-server model is presented in Fig.1. The left block represents the structure of client side, while the right block represents the structure of the server side. The client side is responsible for recording the speech voice and transmitting the speech data to the server side. The server side is responsible for speech recognition. With proper network bandwidth and protocols, the speech features and recognition results are transmitted between clients and servers via the public switched telephone network (PSTN) and Internet.

[0019] The client-server approach has many benefits as follows:

- Mel Frequency Capstral Coefficents (MFCC) is a common set of feature extractors used by many state-of-the-art Hidden Markov Model (HMM) based automated speech recognition (ASR) systems. The client/server DSR model can be implemented in many types of client systems.
- Speech feature extractions efficiently carry the information which the speech recognition function needs. Special feature extractions can be represented in a data size that is much smaller than pure voice data.

[0020] Fig.8 schematically depicts an example of the network environment in which the

architecture for the DSR system development platform of the present invention may be applied. As shown in Fig.8, DSR clients 810, 812, and 814 can access documents from Web server 816 via Internet 820, particularly via the World-Wide Web ("the Web"). The Web is a well-known collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet 820. DSR clients 810, 812, or 814 may be personal computers or various mobile computing devices, such as personal digital assistants or wireless telephones. Personal digital assistants (PDA) are commonly known hand-held computers that can be used to store various personal information including, but not limited to, contact information, calendar information, etc. Such information can be downloaded from other computer systems, or can be inputted by way of a stylus and pressure sensitive screen of the conventional PDA. Examples of PDA's include the PALM (trademark) computer of 3Com Corporation, and the Microsoft CE (trademark) computers, which are each available from a variety of well-known vendors. A user operating a mobile computing device such as a cordless handset, dual-mode cordless handset, PDA, or portable laptop computer generates control commands to access the Internet 820. The control commands may consist of digitally encoded data, DTMF, or voice commands. These control commands are transmitted to the DSR gateway 818. The DSR gateway 818 processes the control command (including performing speech recognition) from the mobile computing device and transmits requests to the Web server 816 via the Internet 820. In response to a request, the Web server 816 sends documents to the DSR gateway 818. Then, DSR gateway 818 consolidates display contents from the document received from Web Server 816 according to a speech recognition result produced by DSR gateway 818. The DSR gateway then sends the

consolidated display contents to the DSR client from which the command was received.

[0021] Fig.2 schematically depicts one embodiment of a comprehensive, practical, and efficient architecture of a distributed speech recognition client and server development platform. As shown, the embodiment of the DSR client and server development platform 10 includes: DSR client development module 100, DSR gateway development module 200 and DSR document server 300. DSR client module 100 captures speech, extracts speech features, interprets markup contents, displays information content on a client display screen, and communicates with DSR gateway module 200. DSR gateway module 200 receives a tagged document from DSR document server 300. In the preferred embodiment, the tagged document can be represented as a conventional Extensible Mark-up Language (XML) document with tags added to assist the speech recognition function. In one embodiment, these XML documents modified to support DSR are denoted DSRML documents. The DSRML documents contain XML elements augmented to support DSR. These elements are denoted DSRML elements. The DSR Gateway module 200 interprets the DSRML elements, dynamically generates grammar from the DSRML elements, controls display content navigation by speech recognition, and communicates with DSR client module 100 and DSR document server 300. DSR document server 300 processes requests from DSR gateway module 200 and produces DSRML documents in response.

[0022] Based on such architecture of one embodiment of the present invention, the working scenario for accessing documents by speech through the Internet is provided as Fig. 3.

[0023] As shown in Fig. 3, the procedure flow of steps includes the following:

1) DSR client 100 initiates a DSR session by negotiating with the DSR gateway 200. At the same time, the DSR client 100 sends some information (such as the type of client device and the characteristics of the user's speech, etc.) to DSR gateway 200 and requests the initial DSRML (DSR XML) document.

2) After receiving the DSR session initial information from DSR client 100, DSR gateway 200 sends a DSRML document request to Web Server 816 (Document Server 300) with the help of a DSR DNS (Domain Name) server. Then, Web Server 816 sends back the requested DSRML documents.

3) After receiving the DSRML documents, DSR gateway 200 parses the DSRML documents first, interprets the document content, and compiles all the grammars that the Recognition Engine will need.

4) DSR gateway 200 also generates display content for the DSR client 100 (which includes several cards of information organized as a document). DSR gateway 200 then sends the display content to the DSR client 100. Then, DSR gateway 200 waits for the client user's speech features to be received from DSR client 100 for processing.

5) After receiving the display content from DSR gateway 200, DSR client 100 displays a relevant card of information.

6) DSR client 100 enables the speech Front-End engine and waits for user utterance input.

7) Upon receipt of user utterance input, the DSR client 100 performs a Front-End algorithm, packs the speech feature data, and sends the packets out to DSR gateway 200.

8) After receiving the speech feature data from the DSR client 100, the DSR gateway 200 starts to perform speech recognition and triggers resulting actions, including those described below.

9) If the recognition result generated by DSR gateway 200 means that DSR client 100 should display another information card in the Document, DSR gateway 200 sends an event (event is DSR gateway's 200 response to DSR client's 100 speech feature) to DSR client 100 with the information of the relevant card identifier. Then, DSR client 100 displays the card and the procedure continues from step 6 above.

10) If the recognition result is not decipherable, DSR gateway 200 sends a corresponding event to DSR client 100.

11) If the recognition result means that DSR client 100 needs a new Document, DSR gateway 200 sends a DSRML document request to Web Server 816. After receiving the needed DSRML document, the procedure continues from step 3 above.

[0024] Fig. 4 illustrates the details of the architecture of one embodiment of a comprehensive, practical, and efficient distributed speech recognition client and server development platform, shown in Fig.2 according to one embodiment of the present invention.

[0025] As shown in Fig. 4, DSR client module 100 includes: audio capturer 101, feature extractor (DSR Front End) 102, data wrapper 103 and interpreter 104, each of which performs a part of the function of DSR client 100.

[0026] Audio capturer 101 captures speech. Audio capturer 101 might fulfill some robust

functionality for a demanding environment. Major functions include recording, sampling a speech signal, and performing voice active detection or end-point detection.

[0027] Feature extractor 102 processes the speech data for speech features. Fundamentally, feature extractor 102 fulfills the functionalities of MFCC and vector quantization. Feature extractor 102 typically performs a short-time Fourier analysis and extracts a sequence of observation vectors (or acoustic vectors) X=[x1, x2 … xt]. Many choices exist for the acoustic vectors, but the cepstral coefficients have exhibited the best performance to date. The sequence of acoustic vectors can either be modeled directly or vector-quantized first and then modeled.

[0028] Data wrapper 103: As shown in Fig.4, there are two data wrappers 103, 211 in one embodiment, which are located respectively at DSR server browser 210 of DSR gateway module 200 and DSR client module 100. The communication between DSR client module 100 and DSR gateway module 200 depends on these two data wrappers. Now, we first describe the data wrapper in DSR client module 100. Data wrapper 103 fulfills the wrapper functionalities of connection request, synchronization of events, and speech feature data transmission. Specifically, data wrapper 103 provides among the following functions: adjusting transmission control conditions according to transmission parameters; implementing the function of a TCP client; receiving DSRML data and sending an initial connect request; controlling whether or not to add Quality of Service (QoS) headers to packets according to a QoS control parameter; compressing speech features; constructing frames; adding a cyclic redundancy check (CRC) header to every frame-pair; constructing multi-frames; adding a CRC header to every multi-frame; and making UDP wrap to sending data and making TCP wrap to sending data. The interpreter 104 transfers

the speech feature streams output from the feature extractor 102 to data wrapper 103. The interpreter 104 also receives documents and events sent from the data wrapper 103 without detailed knowledge of the transmission of this data.

[0029] Interpreter 104 allocates tasks to the audio capturer 101, speech feature extractor 102, and data wrapper 103, which capture, process speech input data for features, and communicate with server browser 211 of DSR gateway 200. The interpreter 104 also receives display content. The display content is organized as a document-cards style. Each document contains several information cards. A user can navigate through the cards, cause the displaying of the contents of each, and cause advancement on to another card. The navigation through the document cards is controlled by the speech recognition results generated in DSR gateway module 200 through events.

[0030] The respective components of DSR client module 100 of one embodiment of the invention were described above. The following section provides a detailed description of DSR gateway module 200.

[0031] As shown in Fig.4, a major function of the DSR gateway module 200 is to get a DSRML document from Document Server 300, to interpret the DSRML elements of the document, to communicate with DSR client module 100 through a wired or wireless network, to dynamically generate grammar from the DSRML elements, and to control display content navigation by speech recognition. Fundamentally, DSR gateway 200 functionality can be separated into three parts:

server browser 210, DNS server 220 and utility platform 230. It will be apparent to one of

ordinary skill in the art that other equivalent functionality groupings are possible.

[0032] Server browser 210 interacts with DSR client module 100 and utility platform 230. Server browser 210 can be separated into three key components: data wrapper 211, HTTP wrapper 212 and interpreter 213.

[0033] When server browser 210 receives the document content from DSR document server 300 through HTTP wrapper 212, server browser 210 calls interpreter 213 to parse the content and to allocate various tasks to utility platform 230. These various tasks include generating display content for the client, and grammar generation.

[0034] When display content has been generated in a document-cards style as described above and the content has been sent to server browser 210, server browser 210 sends the content to DSR client module 100 through data wrapper 211, 103. When server browser 210 receives the speech features from DSR client module 100, server browser 210 forwards the speech features and allocates one more tasks to utility platform 230 which performs speech recognition based on speech features and builds a related grammar.

[0035] After speech recognition, utility platform 230 sends speech recognition results to server browser 210. Finally, according to the speech recognition results, server browser 210 sends a new display card identifier or a new document to DSR client module 100.

[0036] The respective components of server browser 210 are described in more detail below. Server browser 210 includes data wrapper 211, HTTP wrapper 212, and Interpreter 213.

[0037] Data wrapper 211 is similar to the data wrapper 103 in DSR client 100. Data wrapper 211 fulfills the jobs of receiving speech features from and sending display contents and card

identifiers to DSR client 100. Specifically, data wrapper 211 provides among the following functions: adjusting transmission control conditions according to transmission parameters; implementing the function of the TCP server; sending DSRML data and receiving an initial connection request; controlling whether or not to decompose a QoS header of received data packets according to a QoS control parameter; performing multi-frame synchronization; performing error detection and mitigation; performing feature decompression; receiving UDP packets and unwrapping them into proper formats, and receiving TCP packets and unwrapping them into proper formats.

[0038] The HTTP wrapper 212 is responsible for DSRML document requests and document transmission between DSR document server 300 and DSR server browser 210. This transmission is implemented using a standard HTTP conversation in one embodiment.

[0039] Interpreter 213 is the parser of DSRML. After receiving a DSRML document from DSR document server 300, the interpreter 213 will parse the elements of a DSRML document, determine the syntax of the DSRML document, consolidate the information for the server browser 210, and assign various tasks to utility platform 230.

[0040] The server browser 210, according to one embodiment of the invention was described above. The other two parts of the DSR gateway module 200 are described below. These other two parts include the DNS server 220 and utility platform 230.

[0041] The DNS server 220 is implemented within DSR gateway module 200. Using DNS Server 220, HTTP wrapper 212 of DSR server browser 210 can easily and rapidly get domain name interpretation service. The DNS server 220 caches the IP addresses of DSRML document

servers while getting domain name interpretation service from the external DNS servers.

[0042] The utility platform 230 is controlled by server browser 210, and fulfills most of the low-level system jobs. Utility platform 230 can be separated into seven parts: utility platform shell 231, dynamic grammar builder 232, transmission controller 233, display content generator 234, DSR speech recognition engine 235, Text to Speech (TTS) engine 236, and workload balance controller 237.

[0043] The utility platform shell 231 receives the various tasks from server browser 210, allocates the tasks to specific components of utility platform 230, and harmonizes the operation of all utility platform 230 components. Finally, utility platform 231 communicates the results to server browser 210.

[0044] The dynamic grammar builder 232 processes grammar from the speech input. In a DSR application, each dialog has its own grammar, which specifies the allowable input word. Usually, the documents provide grammar as the text format of keywords without garbage from natural speaking. Further, the speech recognition engine needs the compiled binary grammar to process the speech recognition. In particular, the dynamic grammar builder 232 performs among the following tasks:

- extracting grammar script and keywords from documents.
- adding garbage words for natural speaking input.
- compiling the modified grammar to binary format for DSR speech recognition.
- putting the binary format grammar files in cache.

[0045] The grammar builder 232 could be implemented as a tool in speech recognition engine

235. These two components should be supplied and upgraded together. But, there are many benefits if these components are separated into two components as described herein. These benefits include:

- The grammar can be only compiled once and saved in cache for multiple users' threads;
- Besides compiling for binary, grammar builder 232 needs to deal with DSRML, cache, etc. Separate components can facilitate architecture design, implementation, and system management.

[0046] Display content generator 234 consolidates the display contents in DSRML documents and organizes the content into a document-card model like WML, which matches with the dialog style in DSRML document. In this manner, each dialog just matches one card with a corresponding card identifier. DSRML documents enable the interaction between the client and server using a speech input approach. The DSRML document contains the grammar script for speech recognition and corresponding display information.

[0047] The display contents include DSRML script and other external resources such as pictures and grammars. This content is packed together and sent to the client through the network.

[0048] The transmission controller 233 mainly defines the communication model between DSR client module 100 and server browser 210 of DSR gateway module 200. This component enables the use of diverse networks and protocols. The main functions of this component are to detect client type, to balance network load, and to adjust the transmission performance. Some control parameters are sent to the client at the session start. These parameters only need to be sent once. The other parameters are sent during the session, when network status changes. The

control parameters might be, for example:

- CRC: whether the transmission needs CRC, and what kind of CRC to use
- Bandwidth: what are the bandwidth requirements for the feature transmission
- Frame number: according to a particular network status, the proper number of frames to form a multi-frame according to network status.
- QoS related parameters: whether a QoS header should be wrapped. Specification of the kind of QoS parameters.

[0049] These control parameters can be sent to clients in a special defined markup language format by event. For instance,

[0050] The parameters sent at session start could include, for example:

<event type="transmission">

< crc> </crc>

<QoS> </QoS>

</event>

[0051] The parameters sent during the session could include, for example:

<event type="transmission">

< framnum> 16 </framnum>

<bandwidth> </bandwidth>

</event>

[0052] Workload balance controller 237 balances processing and resource needs of the other components of the utility platform 230. DSR gateway 200 is a very high power consuming platform. If the capacity requirements for current users is high, it can be beneficial to balance the workload using clustering servers. In the workload balance controller 237 of one embodiment, five sub-controllers are defined,

- Sub-controller 1: control the balance for the clustering servers of the dynamic grammar builder 232.
- Sub-controller 2: control the balance for the clustering servers of the Transmission Controller 233.
- Sub-controller 3: control the balance for the clustering servers of the display content generator 234.
- Sub-controller 4: control the balance for the clustering servers of the DSR speech recognition engine 235.
- Sub-controller 5: control the balance for the clustering servers of the TTS engine 236.

[0053] The DSR speech recognition engine 235 does the work of speech recognition from speech feature extractions. As speech recognition engine 235 needs binary format grammar, the dynamic grammar builder 232 must be called first.

[0054] Text to Speech (TTS) engine 236 expands the system to support text input. Sometimes, the display approach in audio format is efficient and user preferred. As such, Text to Speech

technology is supported in the present invention.

[0055] The components of one embodiment of a comprehensive, practical, and efficient architecture for a distributed speech recognition client and server development platform was described above by reference to Fig. 4. In addition to the respective components of the system development platform, the synchronization process for DSR client 100 and gateway 200, and the definition for a DSR markup language are detailed below.

[0056] Figures 5-7 illustrate the synchronization process for DSR client 100 and gateway 200 used in one embodiment of the present invention.

[0057] In the DSR development platform of the present invention, events are used to control all the interactions between DSR client module 100 and DSR gateway module 200. Two fundamental types of events are defined for the synchronization between DSR client module 100 and DSR gateway module 200. The events include:

- system synchronization events (for normal and abnormal cases).
- content synchronization events (only for abnormal cases).

System Synchronization Events

[0058] System synchronization events are defined to inform the client of what has happened and of what to do next. The system synchronization events are sent from gateway 200 to client 100 in response to triggers or circumstances including the following:

- client doesn't respond within the timeout interval.

- client doesn't respond intelligibly.
- client requests help.
- inform the client to move on to next dialog.
- error occurred (such as semantic error in the document).
- inform the client to exit the interaction.

[0059] According to the above triggers or circumstances, the corresponding types of system synchronization events include the following:

> The system synchronization event types in normal cases:

- Ready: The server is ready to receive input.
- Forward: The message for the next display card ID.

> The system synchronization event types in abnormal cases:

- Noinput: Default message.
- Nomatch: Default message.
- Help: Default message.
- Error: Default error message. In this case, the gateway 200 will exit.
- Exit: Default message. In the case, the gateway 200 will exit.

[0060] The system synchronization events are generated at the gateway 200 and organized as markup language format. For example, when the gateway 200 finishes the recognition of one input, the gateway 200 will transit to a next dialog and inform the client 100 to show the corresponding next card. Thus, the "forward" synchronization type of event will be sent to client 100, and the related display card ID will be sent through a <message> element. An example of this type of event is as follows:

```
<event type="forward">
        <message> #city    </message>
</event>
```

<u>Content Synchronization Events</u>

[0061] Content synchronization events are used to deal with the abnormal cases according to the procedure pre-defined in DSRML script to cope with the abnormality. These types of content synchronization events include:

- Help: The messages for the help for which the user asked.
- Noinput: The message for no response within the timeout interval.
- Nomatch: The message for no correct recognition result.
- Error: The message for some other error related to application execution.

[0062] The content synchronization events are interpreted by the gateway 200. For instance, in the case of a wrong recognition result during the interaction, the gateway 200 may transfer to another dialog and inform the client to show the related card. Thus, based on the content synchronization event, the system synchronization event “forward” will be generated and sent to the client 100. Following are example scripts.

[0063] The script of the nomatch content synchronization event that will be interpreted by the gateway 200 as follows:

```
<event type=”nomatch”>
<goto href=”#card_id”/>
</event>
```

[0064] The script of the nomatch system synchronization event generated by the gateway 200 according to the script of a content synchronization event as follows:

```
<event type=”forward”>
<message> #card_id </message>
</event>
```

[0065] Figures 5-7 illustrate the synchronization process between DSR client module and DSR gateway module.

[0066] As shown in Fig. 5, during the setup procedure, when initiating a DSR session from the client module 100, the gateway module 200 receives a handshaking signal and sends a DSRML request to the document server 300 with the help of DSR DNS server 220. After getting the DSRML document from the document server 300, DSR gateway module 200 parses the document and generates display content for the client. In order to inform the DSR client module 100 that DSR gateway module 200 is ready for the user's speech input, DSR gateway module 200 sends the normal event "Ready" to DSR client module 100 with the related card identifier.

[0067] As shown in Fig.6, during the communication procedure, if speech features are received normally, DSR gateway module 200 generates and sends the normal event (Forward) according to the speech recognition result. If speech features cannot be received normally, such as the result of a timeout, DSR gateway module 200 generates an abnormal system synchronization event.

[0068] Fig.7 illustrates the details for generating and sending events from the DSR gateway module 200. As shown, the handling of both normal and abnormal events is provided.

[0069] In the DSR application system of the present invention, the interaction model between client and gateway is voice and visual through heterogeneous networks. As such, the definition of the DSR markup language that provides voice-visual interaction between clients and the gateway is important. An embodiment of this DSRML definition of the present invention is provided below. In general, the DSRML should provide functionality to recognize spoken input, navigate among related document-cards, and provide speech synthesis for displaying content in audio format.

[0070] The DSRML language of one embodiment of the present invention provides a means for

collecting spoken input, assigning the input to document-defined request variables, and making decisions as to how to execute the DSRML documents based on the user's input. A document may be linked to other documents through Universal Resource Identifiers (URIs).

The Consideration of Client Device Types

[0071] In one embodiment, DSRML is designed to meet the constraints of a wide range of small, narrowband client devices. These client devices are primarily characterized in some of the following ways:

[0072] Display size - small screen size and low resolution. A small mobile device such as a phone may only have a few lines of textual display, each line containing 8-12 characters.

[0073] Input devices – voice input plus other limited, or special-purpose input methods. The phone typically has a numeric keypad and a few additional function-specific keys. The more sophisticated device may have software-programmable buttons, but may not have a mouse or other pointing device.

[0074] Computational resources - low power CPU and small memory size; often limited by power constraints.

[0075] Narrowband network connectivity - low bandwidth and high latency. Devices with 300bps to 10kbps network connections and 5-10 second round-trip latency are not uncommon.

The Consideration of the Definition for DSR Markup Language

[0076] A DSRML document forms a conversational finite state machine. The user is always in

one conversational state, or dialog, at a time. Each dialog determines the next dialog to which the user transitions.

Dialogs

[0077] In one embodiment of the present invention, there are two kinds of dialogs: link and submit. If the dialog's type attribute is "submit", usually it has a *<submit>* element. The dialog is used to collect values for each of several *<input>* item variables.

[0078] The <dialog> may specify a grammar that defines the allowable word for the input. The user must provide values for *<input>* variables before proceeding to the *<submit>* element in the dialog.

[0079] Another dialog type is "link". In this dialog, there is no specified *<input>* element; but there are links in the display contents. The dialog is used to transit to another dialog through the links. Each link may also specify a grammar.

Application

[0080] An application is a set of documents sharing the same application root document. Whenever the user interacts with a document in an application, its application root document remains loaded while the user is navigating between other documents in the same application. The root document is unloaded when the user navigates to a document that is not in the application. While it is loaded, the application root document's grammars can be set to remain active during the application.

Grammar

[0081] Each dialog has one or more speech grammars associated with it. In server browser directed applications, each dialog's grammars are active only when the user is in that dialog. Both the <*input*> element and the <*link*> element need at least one grammar.

Display

[0082] In a typical DSRML application, the documents contain text and pictures as the display contents. The display content in each dialog is organized as a card showing in the mobile client. Each dialog corresponds to one card, and vice versa. There is a specific element named "display" containing all the visual contents.

Event

[0083] In a typical DSRML application, the system may generate events, including nomatch, time expiration, and user requested help. By default, the system handles all these events, not needing to define them in a document. But, if we have defined them in a DSRML document as content synchronization events, the defined element, such as <*event*> and/or <*help*>, will be executed, for example, sending a warning message to the client.

[0084] The DSR platform of the present invention provides several advantages over the prior art.

Specifically, software vendors can develop better components and tools for the platform using the architecture definition provided herein. Client manufacturers can use the components in the client platform to independently develop DSR handheld devices; Internet Service Providers (ISP) or telecom operators can use the definition and develop components for the server platform to integrate with the DSR gateway; and ICP (Internet content provider) can use the DSR-XML definition and edit tools in the platform to develop DSR service content.

[0085] In one aspect of the present invention, a distributed speech recognition (DSR) system development platform based on client-server framework, includes:

- DSR client module for capturing speech, extracting speech features, interpreting markup contents and display content for presentation on a display screen, and communicating with a DSR gateway module;

- DSR gateway module for receiving a markup document from a DSR document server, interpreting the tag elements of the markup document, dynamically generating grammar from the markup document, controlling display content navigation by speech recognition, and communicating with the DSR client module and a DSR document server; and

- DSR document server for processing requests from the DSR gateway module and producing DSRML documents in response.

[0086] In another aspect of the invention, a method for accessing documents by speech in a

distributed speech recognition system based on the above described platform, includes the components:

- DSR client performing front-end processing and sending the speech feature data to a DSR gateway;
- after receiving the speech feature data from the DSR client, the DSR gateway performs speech recognition and various other tasks including:
  - if the speech recognition result means that the DSR client should display another component of the current document, the DSR gateway will send an event to the DSR client with the identifying information of related component, then the DSR client will display this component of the current document;
  - if the speech recognition result is meaningless or undecipherable, the DSR gateway sends a corresponding event to the DSR client; and
  - if the speech recognition result means that the DSR client needs a new document, the DSR gateway sends a DSRML request to the DSR document server; after receiving the needed DSRML document, the DSR gateway parses the DSRML document, compiles all the grammars that the speech recognition engine needs, generates display content for client, and then sends the display content to the DSR client.

[0087] Thus, an inventive DSR client and server development platform including a DSR client

module, DSR gateway module and DSR document server is disclosed. The scope of protection of the claims set forth below is not intended to be limited to the particulars described in connection with the detailed description of presently preferred embodiments. Accordingly, the present invention provides a comprehensive, practical, and efficient architecture for a DSR client and server development platform.